This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

(19) BUNDESREPUBLIK DEUTSCHLAND

(12) **Patentschrift**

(10) **DE 37 33 038 C 2**

(51) Int. Cl.[5]: H 04 N 7/137

DEUTSCHES PATENTAMT

(21) Aktenzeichen: P 37 33 038.1-31
(22) Anmeldetag: 30. 9. 87
(43) Offenlegungstag: 20. 4. 89
(45) Veröffentlichungstag der Patenterteilung: 5. 1. 94

Innerhalb von 3 Monaten nach Veröffentlichung der Erteilung kann Einspruch erhoben werden

(73) Patentinhaber:
Siemens AG, 1000 Berlin und 8000 München, DE

(72) Erfinder:
Strobach, Peter, Dr.-Ing., 8391 Röhrnbach, DE

(56) Für die Beurteilung der Patentfähigkeit in Betracht gezogene Druckschriften:

WO 86 04 757
IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAM I-7, No.3, May 1985, S.284-289;

(54) Verfahren und Schaltungsanordnung zur Bilddatenreduktion für digitale Fernsehsignale

DE 37 33 038 C 2

DE 37 33 038 C 2

Beschreibung

Die vorliegende Erfindung bezieht sich auf ein Verfahren zur Bilddatenreduktion für digitale Fernsehsignale, wobei in einer Vorverarbeitung eine Differenzbildung zwischen einem Signal, das zu einem Zeitpunkt t−1 erzeugt und in einem Bildspeicher abgelegt wurde, und einem Signal, das zu einem Zeitpunkt t entsteht, durchgeführt wird, so daß durch diese Vorverarbeitung ein Bild-zu-Bild-Prädiktionsfehlersignal gebildet wird, wobei das derart gebildete Bild-zu-Bild-Prädiktionsfehlersignal in Form eines Fehlerbildes zu dessen Übertragung zu codieren ist.

In künftigen Bildtelefonnetzen im Rahmen des Schmalband ISDN wird es erforderlich sein, Videosignale (Sprecherszenen mit eingeschränktem Bewegtanteil) bei einer vorgegebenen Kanalrate von nur 64 kbit/s zu übertragen. Dies entspricht einer Datenkompression um etwa den Faktor 110, was extreme Anforderungen an den zu verwendenden Bildsignal-Codec stellt.

Die auf dem Markt erhältlichen Codecs, welche den genannten Anforderungen genügen, haben etwa die mechanischen Abmessungen einer sog. MICROVAX, eine Leistungsaufnahme von 1 kW und einen Preis von mehreren zehntausend Dollar.

Die geplanten Bildtelefonnetze mit den zugehörigen Endgeräten werden sich jedoch nur dann in breitem Rahmen durchsetzen und wirtschaftlich interessant sein, wenn es gelingt, mechanische Abmessungen, Leistungsaufnahme und Preis der neuen Endgeräte mit den zugehörigen Bildsignal-Codecs im Rahmen der heute gebräuchlichen Fernsprech-Endgeräte zu halten. Betrachtet man den Realisierungsaufwand für die bekannten Verfahren zur Bilddatenkompression, so ist abzusehen, daß dieses Ziel selbst bei höchst optimistischer Einschätzung der Entwicklung zukünftiger Integrationsdichten im Falle des benötigten Bildsignal-Codecs nicht erreicht werden kann.

Der vorliegenden Erfindung liegt die Aufgabe zugrunde, ein neues Verfahren und eine Schaltungsanordnung zur Bilddatenkompression zu schaffen, dessen Funktionsprinzip eine Verbesserung der Bildqualität gegenüber bekannten Verfahren und Schaltungsanordnungen ermöglichen soll.

Diese Aufgabe wird durch ein Verfahren mit Merkmalen nach Anspruch 1 gelöst.

Im folgenden wird die Erfindung anhand mehrerer Figuren im einzelnen beschrieben.

Fig. 1 zeigt ein Prinzipschaltbild eines Szenencodierers mit bewegungskompensierter zeitlicher DPCM.

Fig. 2 zeigt ein aktuelles Bild aus einer Testszene "SIMPLE".

Fig. 3 zeigt das zeitliche Vorgängerbild zum Bild gemäß **Fig.** 2.

Fig. 4 zeigt ein bewegungskompensiertes Differenzbild.

Fig. 5 zeigt eine schematische Darstellung mit einer Objektrandkurve mit beschreibendem "Quadtree".

Fig. 6 zeigt schematisch ein Grundelement (Hauptquadrat) eines "Quadtree" mit den vier Unterquadraten I, II, III, IV.

Fig. 7 zeigt ein Blockschaltbild eines QSDPCM-Coders.

Fig. 8 zeigt ein Blockschaltbild eines QSDPCM-Decoders.

Fig. 9 zeigt eine Darstellung verschiedener Architektur-Grundelemente.

Fig. 10 zeigt ein Blockschaltbild eines sog. Mean-Prozessors.

Fig. 11 zeigt ein Blockschaltbild einer QSDPCM-CPU.

Fig. 12 zeigt eine schematische Darstellung des sukzessiven Zusammenfassens von Teilgebieten durch die QSDPCM-EPU **Fig.** 11.

Fig. 13 zeigt schematisch einen Quantisierer und dessen Quantisiererkennlinie.

Fig. 14 zeigt das Blockschaltbild einer Bitabzähl-Logik.

Fig. 15 zeigt schematisch die Organisation eines Bildspeichers.

Fig. 16 zeigt ein Zeitdiagramm für eine variable Codierzeit.

Fig. 17 zeigt ein Diagramm des Pufferfüllstandes eines Pufferspeichers für das in **Fig.** 16 gezeigte Beispiel.

Fig. 18 zeigt ein Diagramm betreffend die Anzahl von fortgelassenen Bildern über den übertragenen Bildern für die Sequenz "SIMPLE".

Fig. 19 zeigt ein Diagramm über den Verlauf des "MSE" über den übertragenen Bildern für die Sequenz "SIMPLE".

Fig. 20 zeigt ein Diagramm betreffend die Anzahl der fortgelassenen Bilder über den übertragenen Bildern für eine Sequenz "MISS AMERICA".

Fig. 21 zeigt diagrammartig den Verlauf des "MSE" über den übertragenen Bildern für die Sequenz "MISS AMERICA".

Fig. 22 zeigt ein aktuelles Bild aus der Sequenz "SIMPLE" (codiert).

Fig. 23 zeigt das Vorgängerbild zu dem Bild gemäß **Fig.** 22 aus der Sequenz "SIMPLE" (codiert).

Fig. 24 zeigt eine bildhafte Darstellung einer codierten Interframe-Information (4-fach verstärkt).

Fig. 25 zeigt eine Quadtree-Struktur der Interframe-Information.

Fig. 26 zeigt ein aktuelles Bild aus der Sequenz "MISS AMERICA" (codiert).

Fig. 27 zeigt das Vorgängerbild zu dem Bild gemäß **Fig.** 26 (codiert).

Fig. 28 zeigt eine Darstellung der codierten Interframe-Information (4-fach verstärkt).

Fig. 29 zeigt die Quadtree-Struktur der Interframe-Information.

Fig. 30 zeigt eine Koeffizientenmaske mit Koeffizienten A 1, A 2, A 3, A 4, A 5, A 6.

Fig. 31 zeigt eine spiralförmige Adreßsequenz für eine y-x-Verschiebung.

Die Quadtree-Datenstruktur

Die vorliegende Erfindung geht von einer sog. Quadtree-Struktur aus. Die Quadtree-Datenstruktur erlaubt

die Segmentierung eines zweidimensionalen, ortsdiskreten Gebietes in reguläre (quadratische) Suchgebiete variabler Blockgröße, deren Inhalt jeweils durch gemeinsame Eigenschaften charakterisiert werden kann. Quadtrees ermöglichen somit auch die Beschreibung komplizierter Objektrandkurven, vergl. Fig. 5.

Als Grundelement der Quadtree-Struktur gilt ein Quadrat mit der Kantenlänge k, welches in vier gleichgroße Quadranten I, II, III, IV unterteilt sein kann oder als Einheit verbleibt, vergl. Fig. 6.

Ein Quadtree läßt sich grundsätzlich auf zwei unterschiedliche Arten konstruieren:

1. Top Down—
Der Quadtree entsteht durch hierarchische Unterteilung von Hauptquadraten in jeweils vier Unterquadrate, welche im nächsten Schritt wieder als Hauptquadrate behandelt werden.

2. Bottom Up—
Der Quadtree entsteht durch hierarchisches Zusammfassen von vier Unterquadraten zu jeweils einem gemeisamen Hauptquadrat, welches im nächsten Schritt wieder als Unterquadrat behandelt wird.

Die Entscheidung, ob zusammgefaßt (Bottom Up-Methode) oder unterteilt (Top Down-Methode) werden soll, fällt bei beiden Entwurfsmethoden durch folgenden Hypothesentest:

Hypothese 1:
Die vier Unterquadrate I, II, III und IV verhalten sich gleich im Sinne einer vorgegebenen Eigenschaft.

Hypothese 2:
Die vier Unterquadrate verhalten sich unterschiedlich im Sinne derselben Eigenschaft.

Ist die Hypothese 2 erfüllt, so bleiben die vier Unterquadrate erhalten, andernfalls verschmelzen die vier Unterquadrate zu einem Hauptquadrat.

Einzelbildcodierung mit Quadtrees

Die Einzelbildcodierung mit Quadtrees wird bereits seit längerer Zeit verfolgt, vergl. z. B. C. H. Shaffer and H. Samet, "Optimal quadtree constructions algorithms", Comp. Vision, Graphics and Image Processing, Vol. 37, pp. 402—419, 1987; L. L. Jones and S. S. Iyengar, "Space and time efficient virtual quadtrees", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-6, No. 2, pp. 244—247, 1984; A. Klinger and C. R. Dyer, "Experiments on picture representation using regular decomposition", Computer Graphics and Image Processing, Vol. 25, pp. 68—105, 1976; Y. Cohen, M. S. Landy and M. M. Pavel, "Hierarchical coding of binary images", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-7, No. 3, pp. 284—298, 1985; D. S. Scott and S. S. Iyengar, "A new data structure for efficient storing of images", Pattern Recognition Letters, Vol. 3, pp. 211—214, 1985; D. J. Vaisey and A. Gersho, "Variable block-size image coding", Proc. Int. Conf. on ASSP, pp. 1051—1054, Dallas, 1987.

Weiterhin ist aus der WO 86/04757 ein Verfahren zur Datenreduktion für Videosignale bestehend aus einer zeitlichen Folge digitaler Bilder auf der Grundlage der Differenz-Puls-Code-Modulation bekannt, wobei dort auch Differenzbilder als Quadtree-Strukturen codiert werden.

Im folgenden soll auf eine Methode eingegangen werden, bei der der Mittelwert aller Pixel in einem Block zur Konstruktion der Quadtrees herangezogen wird.

Sei

$$M = \sum_{(m)} \sum s(i, j) \quad (3.1)$$

der Mittelwert des Bildsignals {s} im Hauptquadrat, welches durch die Indexmenge

$$i, j \in \{m\} \quad (3.2)$$

bestimmt ist. Ferner hat man noch die Mittelwerte der Unterquadrate M_I, M_{II}, M_{III} und M_{IV} zur Verfügung. Nun erhalten die Hypothesen folgende Gestalt:

Für den Fall

$$(|M/4\text{-}M_I| > \sigma) \vee (|M/4\text{-}M_{II}| > \sigma) \vee (|M/4\text{-}M_{III}| > \sigma) \vee (|M/4\text{-}M_{IV}| > \sigma) = .\text{true.} \quad (3.3)$$

entscheidet man sich für Hypothese 2 (Unterquadrate bleiben erhalten), andernfalls fällt die Entscheidung zu Gunsten der Hypothese 1, wobei σ eine vorgegebene Schwelle ist. Dieses Verfahren verdient die Bezeichnung "Gleichanteilsdekomposition" da das Bild in Gleichanteile verschiedener Flächengröße zerlegt wird. Das Bild läßt sich beliebig genau approximieren, wenn nur die Schwelle σ genügend klein gewählt wird. Das Verfahren erreicht eine hohe Datenkompression in homogenen Bereichen, während detailreiche Gebiete auf feine Quadtrees, und damit auf eine geringe Datenkompression führen. Das Verfahren eignet sich besonders für die Codierung von Bildern mit linienartigen Strukturen, zwischen denen sich weite homogene Flächen erstrecken.

Ein weiterer interessanter Effekt bei diesem Verfahren ist der geringe Rechenaufwand zur Berechnung der

Quadtrees. Wählt man die Bottom Up-Realisierung, so können die benötigten Mittelwerte in jeder Hierarchiestufe des Verfahrens über die Formel

$$M = M_I + M_{II} + M_{III} + M_{IV} \quad (3.4)$$

rekursiv berechnet werden. Der erforderliche Rechenaufwand zur Konstruktion des Quadtrees beträgt daher unabhängig vom Datenmaterial in jedem Fall nur eine Addition pro Pixel, zuzüglich der Entscheidungsoperationen aus Formel (3.3).

Szenencodierung mit Quadtrees — Das ASDPCM-Verfahren

Im vorstehenden wurde die reguläre Gleichanteilsdekomposition als ein mögliches Verfahren zur Einzelbildcodierung betrachtet. Nun stellt sich die Frage, inwieweit sich dieses Prinzip auf die Szenencodierung erweitern läßt und insbesondere, inwieweit die besonderen Eigenschaften der quadtreebasierten Gleichanteilsdekomposition (z. B. hohe Datenkompression bei Linienbildern) vorteilhaft in der Szenencodierung genutzt werden können.

Es ist zunächst festzustellen, daß einzelne Bilder aus einer Szene in hohem Maße statistisch abhängig sind, d. h. nur geringe Anteile eines aktuellen Bildes weisen Veränderungen verglichen mit seinem zeitlichen Vorgängerbild auf. Diese Veränderungen müssen codiert werden (Interframecodierung). Sie sind meist nur die Folge einer Bewegung der aktiven Objekte der Szene. Wenn es nun gelingt, diese Objektbewegungen in zeitlich aufeinanderfolgenden Bildern wenigstens annähernd zu kompensieren, so erhält man ein Differenzbild, welches nur an den Objekträndern signifikante Amplituden aufweist. Zwischen diesen Randlinien erstrecken sich weitgehend homogene Flächen. Es sei an dieser Stelle an die besonders günstigen Datenkompressionseigenschaften der Gleichanteilsdekomposition bei so gearteten Bildern erinnert.

Die Formeln für die Gleichanteilsdekomposition im Szenenfall (Interframemodus) lauten also:

$$M(x,y) = \sum_{(m)} \sum [s_t(i,j) - s_{t-1}(i-x, j-y)] \quad (4.1)$$

wobei $\{s_t\}$ und $\{s_{t-1}\}$ die zeitlich aufeinanderfolgenden Bilder und

$$z = [x, y]^T \quad (4.2)$$

der Bewegungsvektor ist, welcher für ein Gebiet {m} gilt. Die Entscheidungen zur Konstruktion des Quadtrees erhält man nach wie vor durch Anwendung der Beziehung (3.3) mit anschließendem Test.

Das bis jetzt geschilderte Vorgehen würde zunächst ein eigenes Verfahren für die Bewegungskompensation mit anschließender Gleichanteilsdekomposition (z. B. an Stelle einer Transformation) erfordern. Es ist jedoch ersichtlich, daß eine Bewegungskompensation auf einen umso "einfacheren" Quadtree führt, je vollkommener sie ausgeführt wurde. Im Grenzfall, d. h. bei einer perfekten Bewegungskompensation würden die bewegten Bereiche in aufeinanderfolgenden Bildern exakt zur Deckung gebracht, wodurch das Differenzbild überall den Wert Null annimmt. Von den beschreibenden Quadtree bliebe dann nur noch das große Hauptquadrat übrig.

Die Definition der "Einfachheit" eines Quadtree ist also zunächst zweckmäßigerweise die Anzahl seiner Subgebiete.

Da zu erkennen ist, daß eine gute Bewegungskompensation auf einen einfachen Quadtree führen muß, ist es folgerichtig, zu untersuchen, inwieweit sich dieser Schluß auch umkehren läßt. Es stellt sich die Frage, ob es demzufolge auch möglich ist, den optimalen Bewegungsvektor (d. h. die optimale Verschiebung des Bildausschnitts im Vorgängerbild) dadurch zu ermitteln, daß man die Gleichanteilsdekompensation für alle möglichen Verschiebungen x und y eines Suchbereiches durchführt und schließlich diejenige Verschiebung als die geschätzte Bewegung einsetzt, für die die Anzahl der Subgebiete des zugehörigen Quadtree ein Minimum annimmt. In der Tat zeigte sich im Experiment, daß dieses Vorgehen sehr gute Ergebnisse liefert. Die Bewegungen in der rekonstruierten Szene behalten ihren natürlichen Charakter, während die Anzahl der Subgebiete in den Quadtrees durchschnittlich 30% geringer war als bei Anwendung der eingangs erwähnten "Hybridkonzepts" mit Blockmatching und anschließender Gleichanteilsdekompensation. Dies ist eine bedeutende Aussgage, denn schließlich ist die Anzahl der Subgebiete wenigstens annähernd ein Maß für die Anzahl der Bits, welche für die Codierung der Interframe-Information benötigt werden. Diese neue Methode der "Quadtree Structured Difference Pulse Code Modulation" (QSDPCM) ist auch vom Aufwand her zu rechtfertigen. So benötigt man für die Berechnung sämtlicher möglicher Quadtrees in einem Suchbereich nach der Bottom Up-Methode nur n^2 Festkommaaddition, wenn der Suchbereich $n \times n$ Pixelverschiebungen umfaßt. So ergibt sich beispielsweise bei einem Suchbereich von 11×11 ein Rechenaufwand von nur 121 Festkommaadditionen pro Pixel für das QSDPCM-Verfahren.

Nachdem nun mit der Minimierung der Anzahl der Subgebiete eines Quadtrees schon ein sehr gutes (im Sinne einer hohen Datenkompression) neues Fehlermaß für die (implizite) Bewegungsschätzung eingeführt ist, müßte es nun auch möglich sein, in einem letzten Schritt denjenigen Quadtree und zugehörigen Bewegungsvektor zu ermitteln, welcher tatsächlich auf ein Minimum der Anzahl der zu übertragenden Bits führt. Ein solches Verfahren müßte — verglichen mit herkömmlichen Methoden der Bewegungsschätzung — gute Datenkompressionseigenschaften aufweisen, denn es wird ja direkt die Update-Information (in Bits) minimiert. Tatsächlich ist es von der Minimierung der Subgebiete eines Quadtrees nur noch ein weiterer Schritt bis zu einem Verfahren, welches

den Quadtree mit zugehörigem Bewegungsvektor so entwirft, daß die zu übertragende Update(interframe-)Information (in Bits) minimal wird. Dazu wird zunächst das QSDPCM-Verfahren mit der Minimierung der Subgebiete als Fehlermaß an einer typischen Szene durchgeführt. Die sich ergebenden Mittelwerte in den Subgebieten, die Quadtreecodes und auch die geschätzten Bewegungsvektoren werden gespeichert und schließlich optimal (Huffman) codiert. Die sich ergebenden Codetabellen werden schließlich in das Verfahren eingesetzt. Im endgültigen Betrieb kann das Verfahren nun für jede Verschiebung die gesamte Interframe-Information berechnen, indem nicht die Anzahl der Subgebiete, sondern die Anzahl der Bits, welche für die Codierung des jeweiligen Mittelwertes in einem Subgebiet (durch Nachschauen in den Huffmantabellen) akkumuliert wird, zuzüglich der Anzahl der Bits für den Quadtreecode und die Anzahl der Bits zur Codierung des entsprechenden Bewegungsvektors.

Es hat sich herausgestellt, daß durch diese direkte Minimierung der Interframe-Information, wie vorgehend geschildert, noch einmal eine Einsparung an Bits von ca. 20% erreicht werden konnte. Dies erklärt sich daraus, daß nun auch die Statistik der Mittelwerte in den einzelnen Subgebieten optimal im Sinne einer minimalen Datenrate ausgenutzt wird. Der dafür erforderliche Zusatzaufwand besteht nur in einer Adressierungsoperation (Nachschauen in den — sehr kleinen — Huffmantabellen mit Entnahme der Anzahl der Bits für das Codewort und entspechende Aufakkumulation in einem Bitzähler). Dies ist sehr einfach zu realisieren und bedeutet einen praktisch vernachlässigbaren Zusatzaufwand. Tatsächlich können neben den Huffman Codetabellen für den vorgenannten Zweck auch Tabellen gelegt werden, welche die Anzahl der Bits/Codewort enthalten. Bemerkenswert ist noch, daß die Mittelwerte für unterschiedlich große Subgebiete eine unterschiedliche Amplitudenstatistik aufweisen. In kleinen Blöcken ist die Streuung sehr groß, während die Streuung bei wachsender Blockgröße stark abnimmt. Deshalb wird für jede Blockgröße eine eigene Huffmantabelle eingesetzt. Die Huffmantabellen sind sehr kurz. So ergibt sich bei einer Blockgröße von 16 x 16 beispielsweise nur noch eine Huffmantabelle mit 3 Einträgen. Bemerkenswert ist auch, daß die Statistik der Mittelwerte weitgehend unabhängig vom Szenenmaterial ist, d. h. wurden die Huffman-Tabellen für Szene A entworfen, so sind sie auch für Szene B annähernd optimal. Versuche haben ergeben, daß eine Codeanpassung auf eine neue Szene nur eine Verbesserung von ca. 2% bringt.

Der neue QSDPCM-Codec

Nachdem im Vorstehenden das QSDPCM-Verfahren bereits hergeleitet und erläutert wurde, wird nun seine Anwendung in einem Szenencodec erläutert, der in der Simulation erprobt wurde. Zunächst sei jedoch das zugrundeliegende QSDPCM-Verfahren nochmals in einem Grobflußdiagramm zusammengefaßt. Der QSDPCM-Codec, welcher in **Fig.** 7 gezeigt ist, hat einen Gesamtrealisierungsaufwand von nur ca. 300 Millionen Festkommaadditionen pro Sekunde zuzüglich der Entscheidungsoperationen und Vergleiche und eignet sich aufgrund seiner regelmäßigen Struktur hervorragend für eine VLSI Realisierung auf einem Chip.

Grobflußdiagramm des QSDPCM-Verfahrens

$\forall x \varepsilon$ Suchbereich *do*:

- $\forall y \varepsilon$ Suchbereich *do*:
  1. Konstruiere Quadtree für Verschiebung x,y mit den Beziehungen (4.1), (3.4) und (3.3)
  2. Codiere Quadtree x,y
  3. Codiere Mittelwerte x,y
  4. Codiere Bewegungsvektor x,y
  5. Gesamtfunktion x,y = Quadtreeinformation x,y + Σ Mittelwertinformation $[x,y]$ + Bewegungsvektorinformation $[x,y]$
  6. Suche Minimum $[x,y]$ in der Gesamtinformation $[x,y]$

Ausgabe: 1. Optimalen Bewegungsvektor (codiert)
2. Code des optimalen Quadtree
3. Codierte Mittelwerte für die Subgebiete des optimalen Quadtree

Architektur des QSDPCM-Verfahrens

Nachfolgend wird ein detaillierter Architekturvorschlag für das QSDPCM-Verfahren aufgezeigt. Dies ermöglicht einerseits ein besseres Verfahren des QSDPCM-Verfahrens und andererseits eine genauere Abschätzung des Realisierungsaufwandes für den gesamten auf dem QSDPCM-Verfahren basierenden Codec. Die im folgenden erläuterte Architektur repräsentiert den aktuellen Stand der Simulation und entspricht in allen Details dem Verfahren, mit dem Statistiken und Simulationsergebnisse errechnet wurden.

Fig. 7 zeigt den QSDPCM-Coder, **Fig.** 8 zeigt den zugehörigen Decoder. Die Funktion der darin enthaltenen Blöcke wird im folgenden im einzelnen erläutert. Dazu sind zunächst verschiedene Architektur-Grundelemente (Basiszellen) erforderlich, die in **Fig.** 9 gezeigt und definiert sind. **Fig.** 10 zeigt die Detailschaltung des 2*2 Mean Prozessors. Dieser Prozessor generiert für einen 16*16 Block die 2*2 Mittelwerte der Bilddifferenzen zwischen dem zu codierenden Bild und dem Bildspeicherinhalt und schreibt das Ergebnis dieser Operation in einem Fast

RAM Zwischenpuffer der Größe 64*8 Bit. Diese Operation muß für jede Verschiebung im Suchbereich durchgeführt werden. Die Operation benötigt bei dieser Architektur insgesamt 16 Taktzyklen.

Die weitere Verarbeitung dieser Mittelwerte erfolgt nun in der QSDPCM Central Processing Unit, vergl. Fig. 11. Schritthaltend mit dem Mean-Prozessor müssen hier die in einem Fast RAM bereitgestellten Daten auf weitere Zusammenfaßbarkeit entsprechend dem QSDPCM-Verfahren (vergl. das vorstehende Grobflußdiagramm) überprüft und gegebenenfalls zu größeren durch ihre Mittelwerte hinreichend genau beschreibbaren Gebieten zusammengefaßt werden. Fig. 12 verdeutlicht nochmals die Funktion des QSDPCM-Verfahrens. Im Schritt 1 werden zunächst die 2*2 Blöcke auf weitere Zusammenfaßbarkeit überprüft (16 Taktzyklen). In Schritt 2 werden die 4*4 Blöcke auf Zusammenfaßbarkeit überprüft (4 Taktzyklen) und schließlich wird im 3. Schritt noch die Zusammenfaßbarkeit der 8*8 Blöcke zu einem 16*16 Block (1 Taktzyklus) geprüft. Im Gegensatz zum Mean-Prozessor, welcher mit nur 16 Taktzyklen auskommt, benötigt man bei der QSDPCM-CPU eine Gesamtrechenzeit von 14+4+1=21 Taktzyklen pro verschobenem 16*16 Block. Die Rechenzeit der QSDPCM-CPU ist also bestimmend, da Mean-Prozessor und QSDPCM-Prozessor zeitlich überlappt arbeiten. Das Fast RAM ist doppelt (als Wechselpuffer) ausgelegt. Der Mean-Prozessor hat 21−16=5 Wait-Zyklen. Dazu sei hier noch besonders darauf hingewiesen, daß während der schnellen Suchprozedur kein Rückschreiben von Ergebnissen aus dem Fast RAM in den Bildspeicher erforderlich ist.

Nun bietet sich ein gutes Zusammenspiel zwischen dem maximal berechenbaren Suchbereich und dem Bildinkrement an, wenn man berücksichtigt, daß stark bewegte Bilder immer auf ein höheres Bildinkrement führen. Der Prozessor hat in solchen Fällen sehr viel mehr Zeit und kann größere Suchgebiete abarbeiten. Es ist aber auch zweckmäßig, bei höherem Bildinkrement den Suchbereich zu vergrößern, da sich die Objekte ja dann bereits weiter bewegt haben können. Diese Eigenschaft erhält man bei der vorgeschlagenen Architektur ohne Zusatzaufwand wie folgt: Zunächst muß dafür gesorgt werden, der Motion Vector Generator (Displacement Adressgenerator) einen "Spiralscan" ausgibt, vergl. Fig. 31.

Legt man nun ein Bildformat von 352*288 Pixel (Luminanz)+ 176*288 Pixel (Chrominanz) mit 25 Hz Bildwechselfrequenz zugrunde und wählt man eine synchrone Taktrate von 25 MHz für den Prozessor, so können pro Originalbild für jeden 16*16 Block 80 Verschiebepositionen abgearbeitet werden. Für diese Berechnung wurde eine Anzahl von 594 16*16 Blöcken (CIF inklusive Chrominanz, wie oben bereits angegeben) und eine Bildwechselfrequenz von 25 Hz angenommen. Damit ergeben sich also in Abhängigkeit vom Bildinkrement die folgenden Suchbereiche:

Bildinkrement	Anzahl der Suchschritte	entspricht Suchbereich
1 (25 Hz)	80	+/−4 pixel
2 (12,5 Hz)	160	+/−6 pixel
3 (8,33 Hz)	240	+/−8 pixel
4 (6,25 Hz)	320	+/−9 pixel
5 (5 Hz)	400	+/−10 pixel

Es fällt auf, daß für jedes Bildinkrement die Anzahl der möglichen Suchschritte nicht direkt mit der erforderlichen Anzahl von Suchschritten für einen entsprechenden Suchbereich übereinstimmt. Der äußere "Ring" des Suchbereichs ist also in jedem Fall nicht vollständig besetzt. Hier ist es zweckmäßiger, anstelle des einfachen Spiralscan einen "Lawinenscan" einzusetzen, bei dem die Adressgenerierung in einer Weise vorgenommen wird, daß die noch möglichen Suchpositionen am äußeren Ring des Suchbereichs in etwa äquidistant besetzt sind. Die Realisierung des "Lawinenscan" ist nicht aufwendiger als die des einfachen Spiralscan, da die Adressenfolge zweckmäßigerweise aus einem linear über einen Zähler adressierten PROM abgerufen wird.

Die beiden Blöcke Mean Prozessor (Fig. 10) und QSDPCM-CPU (Fig. 11) beinhalten bereits den gesamten Aufwand an arithmetischen Operationen zur Ausführung des QSDPCM-Verfahrens. Fig. 12 erläutert die Funktion der Quadtree Construction Logic. Der Quadtree wird standardmäßig mit einem Code mit variabler Wortlänge codiert (vergl. ebenfalls Fig. 12).

Fig. 14 zeigt die Bitcount Logik und den Huffman-Coder. Der Huffman-Coder enthält 4 Codetabellen (jeweils 65 Einträge) für die zu codierenden Differenzbildmittelwerte in den 2*2, 4*4 und 16*16 Blöcken, sowie die Huffman-Tabelle für die Codierung der Bewegungsvektoren mit 400 Einträgen. Es wird nur ein Huffman-Coder benötigt, der auf die (langsamen) Tabellen zugreift. Jeder Huffman-Codetabelle ist nun noch eine ebenso große Tabelle zugeordnet, welche als Eintrag die Anzahl der Bits für das entsprechende Codewort enthält. Auf diese Tabellen muß während des Full Search-Suchvorganges schnell zugegriffen werden. Für jeden Verschiebungsvorschlag (Vektorhypothese) im Suchbereich kann somit die Anzahl der Bits, die sich für die Codierung der Interframe-Information ergeben würde, berechnet werden, ohne daß jedoch tatsächlich codiert wird. Die Codierung erfolgt nur für die optimale Verschiebung, d. h. für jene Verschiebung, für die die gesamte Interframe-Information ein Minimum annimmt. Dies ist ein wesentliches Merkmal des QSDPCM-Verfahrens. Der Code für die Interframe-Information eines 16*16 Blocks wird schließlich im Multiplex an den Ausgangspuffer weitergeleitet.

Nach so erfolgter Codierung wird der bearbeitete 16*16 Block im Bildspeicher aktualisiert. Dabei ist es wichtig zu beachten, daß das aktualisierte Bild im Bildspeicher wiederum nur durch seine 2*2 Mittelwerte repräsentiert werden darf. Die Mittelwertbildung muß unbedingt mit dem vom Mean Prozessor auf das Eingangsbild angewendeten Blockraster übereinstimmen, da nur so das Anwachsen von hochfrequenten Störungen im Bildspeicher verhindert wird.

Fig. 15 zeigt die Organisation des Bildspeichers. Beim CIF-Format gilt: NSPA=352, NZEI=288.

Fig. 13 zeigt die Ausführung des Quantisierers für die Blockdifferenzen. Es wird nur eine Aussteuerung von +/—96 zugelassen. Darüber hinausgehende Werte werden begrenzt. Dies ist ein völlig ausreichender Wertebereich, wie Versuche gezeigt haben. Dieser beschränkte Wertebereich wird auch in den Prozessoren bereits genutzt. Hier wird die Pixeldifferenz nicht mit 9 Bit, sondern nur mit 8 Bit dargestellt (Begrenzung bereits in den 4-Input-Addern).

Die konstante Kanalrate am Ausgang des Coders erreicht man über einen Puffer mit variabler Frame-Rate, d. h. der Coder hält die Bildqualität fest und regelt die Datenrate durch Weglassen einer variablen Anzahl von Eingangsbildern. Dieses Konzept führt auf einen äußerst einfachen Pufferregler. Der Pufferspeicher faßt maximal 25 codierte Bilder (1 Sekunde). Dies wird jedoch nur beim Bildaufbau erreicht. Da der Pufferspeicher als echtes FIFO arbeitet, ist die Verzögerung abhängig von der Anzahl der weggelassenen Eingangsbilder und beträgt im günstigsten Fall nur 1/25 Sekunde. Dieser Fall (kein Bild weggelassen) wird beispielsweise bei der Testsequenz "SIMPLE" mehrmals erreicht.

Simulationsergebnisse

Abschließend werden einige Simulationsergebnisse aufgezeigt. **Fig.** 20 zeigt die Anzahl der ausgelassenen Bilder pro übertragenem Bild für die beiden Testsequenzen "SIMPLE" und "MISS AMERICA". In **Fig.** 19 sind die zugehörigen Verläufe des Mean Squared Error (MSE) dargestellt. Es fällt auf, daß bei der Sequenz "MISS AMERICA" mehr Bilder ausgelassen werden müssen als im Falle von "SIMPLE". Ursachen hierfür sind die völlig unterschiedlichen Eigenschaften der beiden Testsequenzen. "SIMPLE" weist einen sehr viel höheren Detailgehalt und auch wesentlich detailreichere bewegte Objekte auf als "MISS AMERICA". Andererseits hat "MISS AMERICA" einen höheren Bewegtanteil verursacht durch ein großes, detailarmes Objekt, das sich zudem kaum vom Hintergrund abhebt. Die bewegungskompensierten Differenzbilder, welche letztendlich die zu codierende Interframe-Information darstellen, weisen im Falle von "SIMPLE" annähernd ideale, sehr hochfrequente Linienstrukturen an den Rändern der bewegten Objekte und an den bewegten Strukturkomponenten auf. Die Testsequenz "SIMPLE" eignet sich deshalb sehr gut für die Codierung mit dem neuen QSDPCM-Verfahren. Weniger gute Ergebnisse wurden bei der Sequenz "MISS AMERICA" erzielt. Da sich hier das bewegte Objekt kaum sichtbar vom Hintergrund abhebt, und auch sonst keine detailreichen bewegten Strukturen auftreten, ist auch das bewegungskompensierte Differenzsignal wesentlich weniger hochfrequent und wesentlich höher örtlich korreliert als im Falle von "SIMPLE". Trotz des niedrigen MSE zeigt das QSDPCM-Verfahren im Falle von "MISS AMERICA" eine schlechtere subjektive Bildqualität in den detail- und kontrastarmen bewegten Gebieten, während Gesicht und Augen der Testperson sehr scharf und natürlich dargestellt werden.

Die vorliegende Erfindung ist nicht allein zur Codierung von digitalen Fernsehsignalen geeignet, sondern beispielsweise auch anwendbar auf die Detektion bewegte Objekte, beispielsweise für Intrusionsschutzanlagen.

Patentanspruch

Verfahren zur Datenreduktion für Videosignale bestehend aus einer zeitlichen Folge digitaler Bilder auf der Grundlage der Differenz-Puls-Code-Modulation mit Bewegungskompensation, wobei

a) Differenzbilder als Quadtree-Strukturen codiert werden,

b) Bewegungsvektoren dadurch bestimmt werden, daß in einem der Bilder für die Differenzbilder Gebietsverschiebungen durchgeführt werden, die Kodierung für jede Gebietsverschiebung vollständig durchgeführt wird, die Zahl der Bits bestimmt wird, die nach der Kodierung entstanden ist und als Bewegungsvektor für eine Gebietsverschiebung derjenige bestimmt wird, für den die Zahl der Bits nach der jeweiligen Kodierung minimal ist.

Hierzu 15 Seite(n) Zeichnungen

Nummer:
Int. Cl.[5]:
Veröffentlichungstag:
DE 37 33 038 C2
H 04 N 7/137
5. Januar 1994

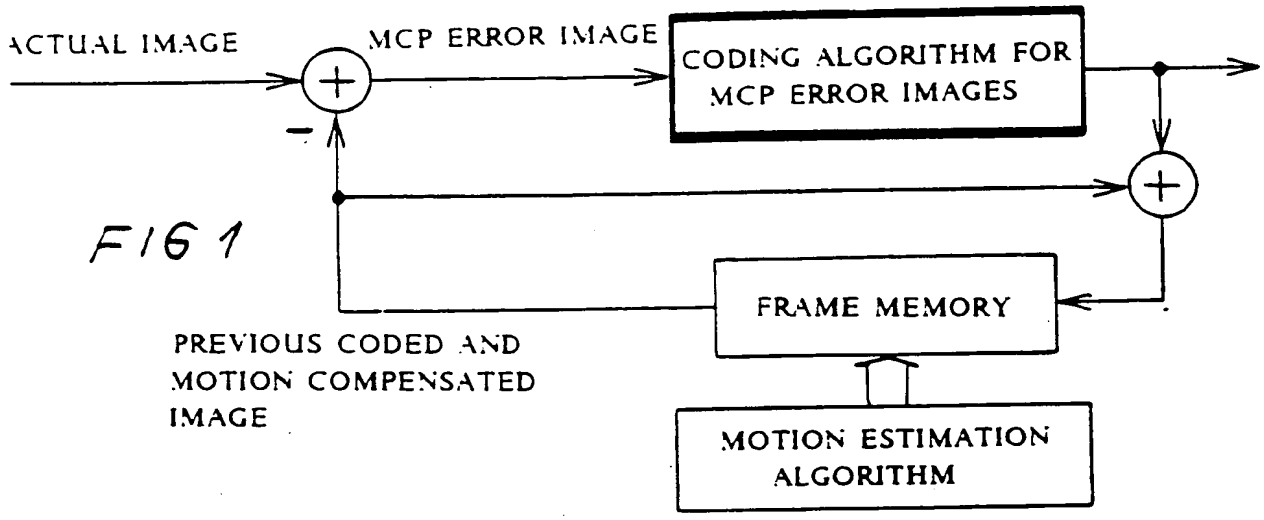


FIG 1

FIG 2

FIG 3

DE 37 33 038 C2
H 04 N 7/137
Veröffentlichungstag: 5. Januar 1994

FIG 4

FIG 5

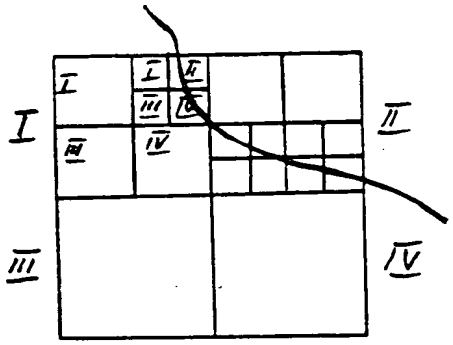
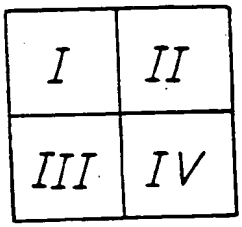


FIG 6



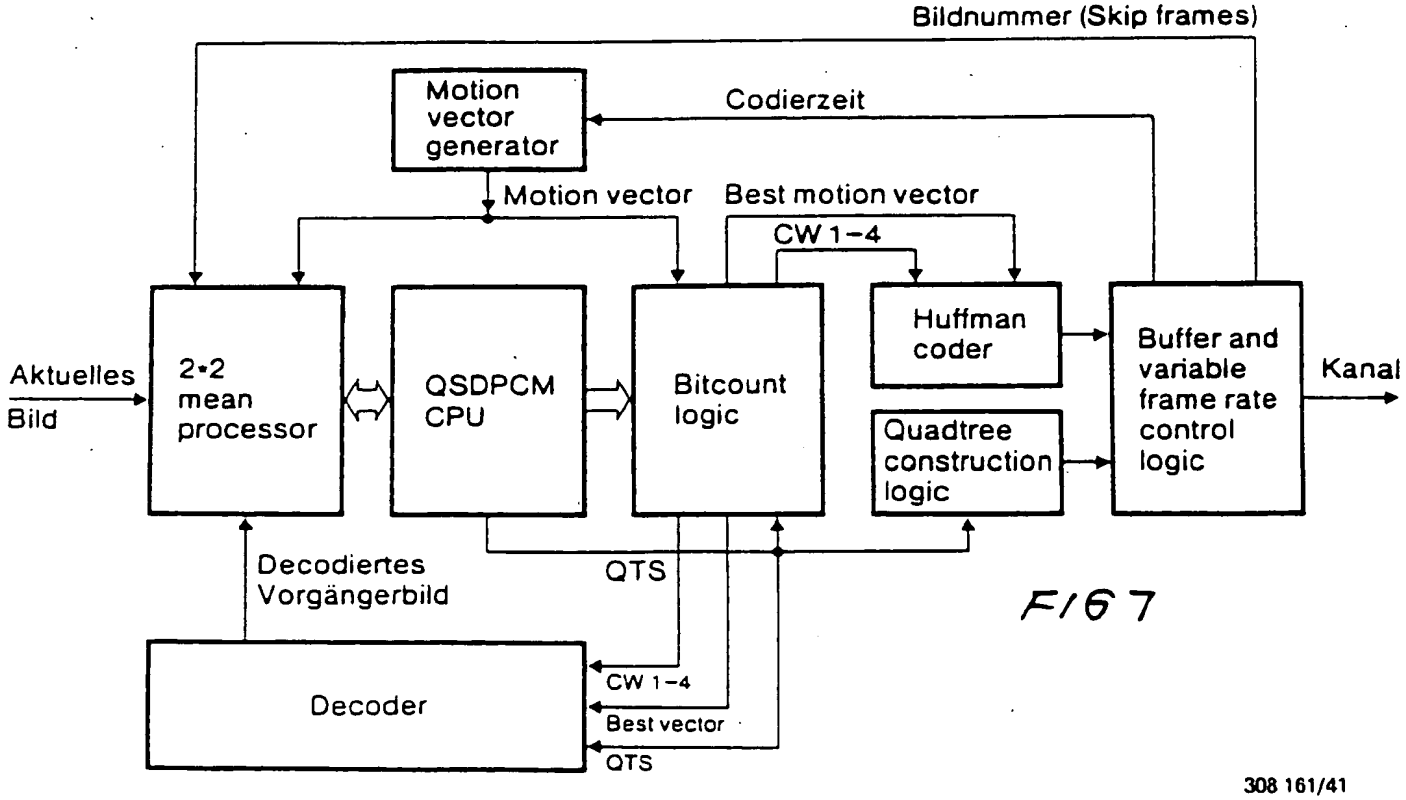


FIG 7

Nummer: DE 37 33 038 C2
Int. Cl.[5]: H 04 N 7/137
Veröffentlichungstag: 5. Januar 1994

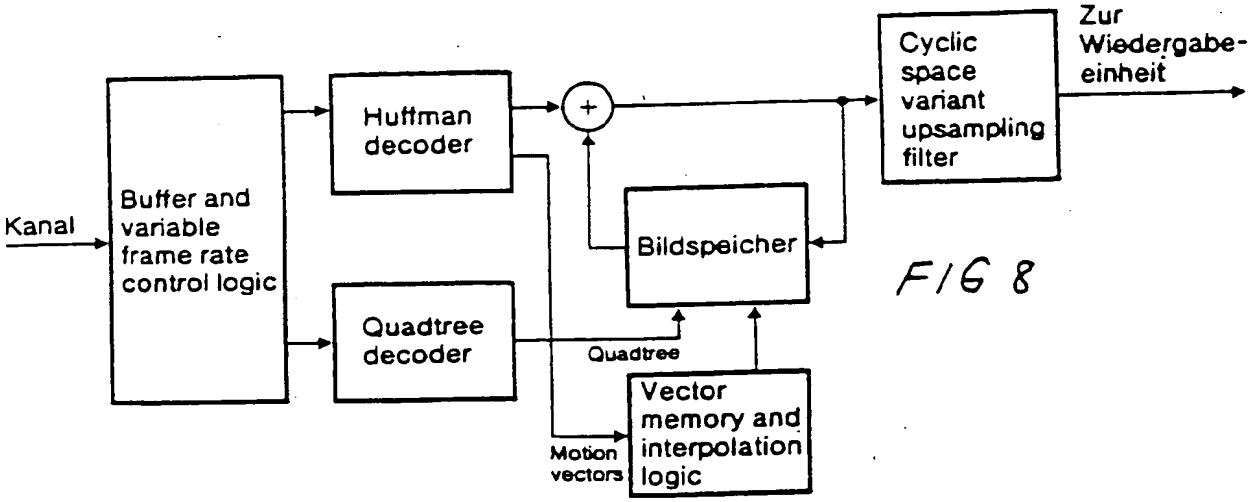


FIG 8

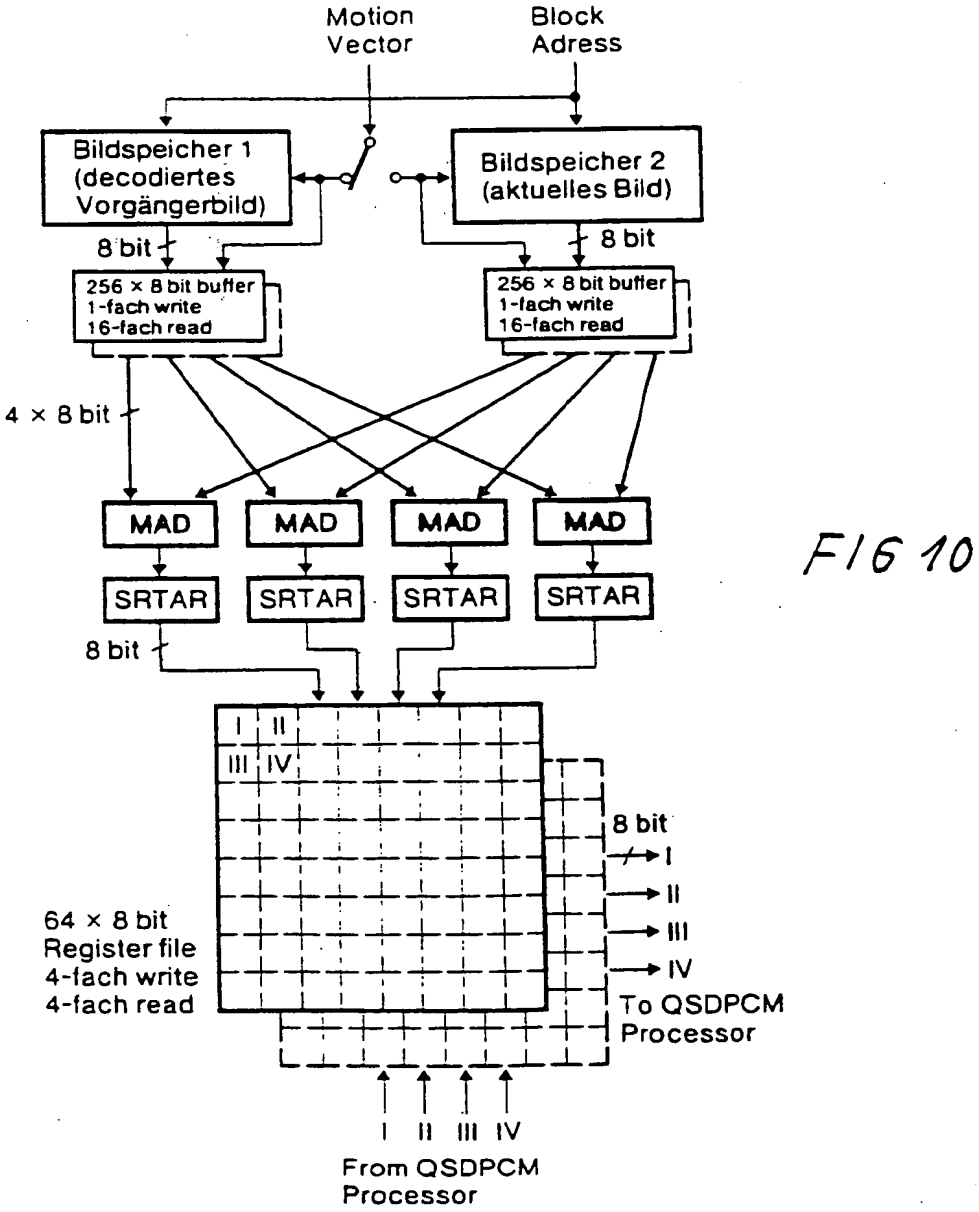
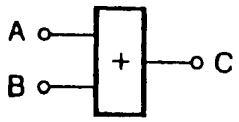


FIG 10

FIG 9

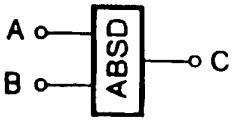
Architektur–Grundelemente

1. Addierer:



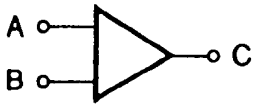
C = A + B

2. Absolute Differenz:



C = |A–B|

3. Komparator:

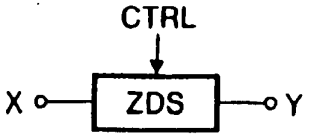


IF (A > B) THEN
C = 1
ELSE
C = 0
ENDIF

4. 2 × Rechtsschieben und Runden:

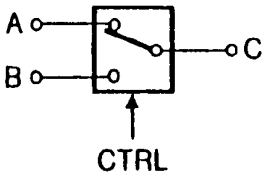
Y = NINT (REAL (X)/4)

5. Daten/Null-Schalter



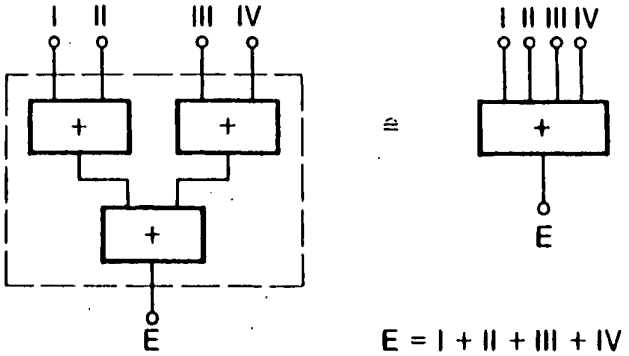
IF (CTRL = 1) THEN
Y = X
ELSE
Y = 0
ENDIF

6. Mulliplexer:



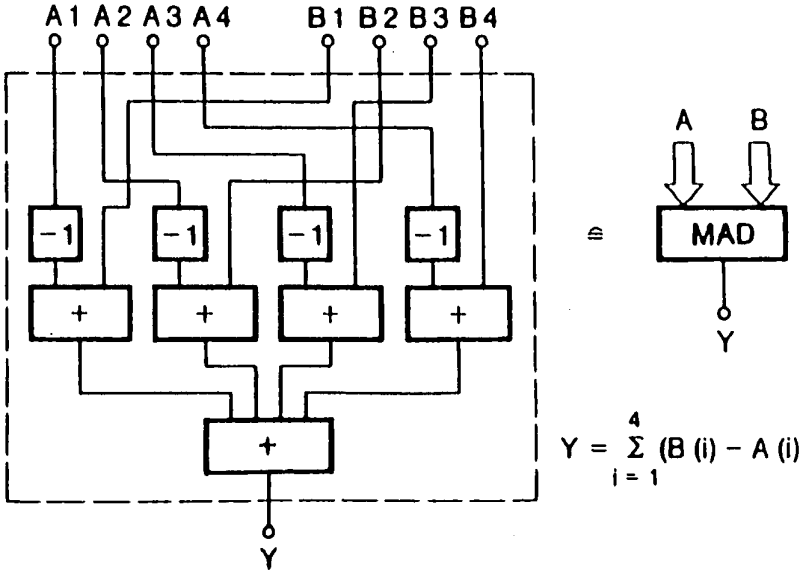
IF (CTRL = 1) THEN
C = B
ELSE
C = A
ENDIF

7. 4-fach Addierer:



E = I + II + III + IV

8. Mean Absolute Difference (MAD) Logik:



$$Y = \sum_{i=1}^{4} (B(i) - A(i))$$

ZEICHNUNGEN SEITE 4

DE 37 33 038 C2
H 04 N 7/137
Veröffentlichungstag: 5. Januar 1994

Nummer: DE 37 33 038 C2
Int. Cl.[5]: H 04 N 7/137
Veröffentlichungstag: 5. Januar 1994

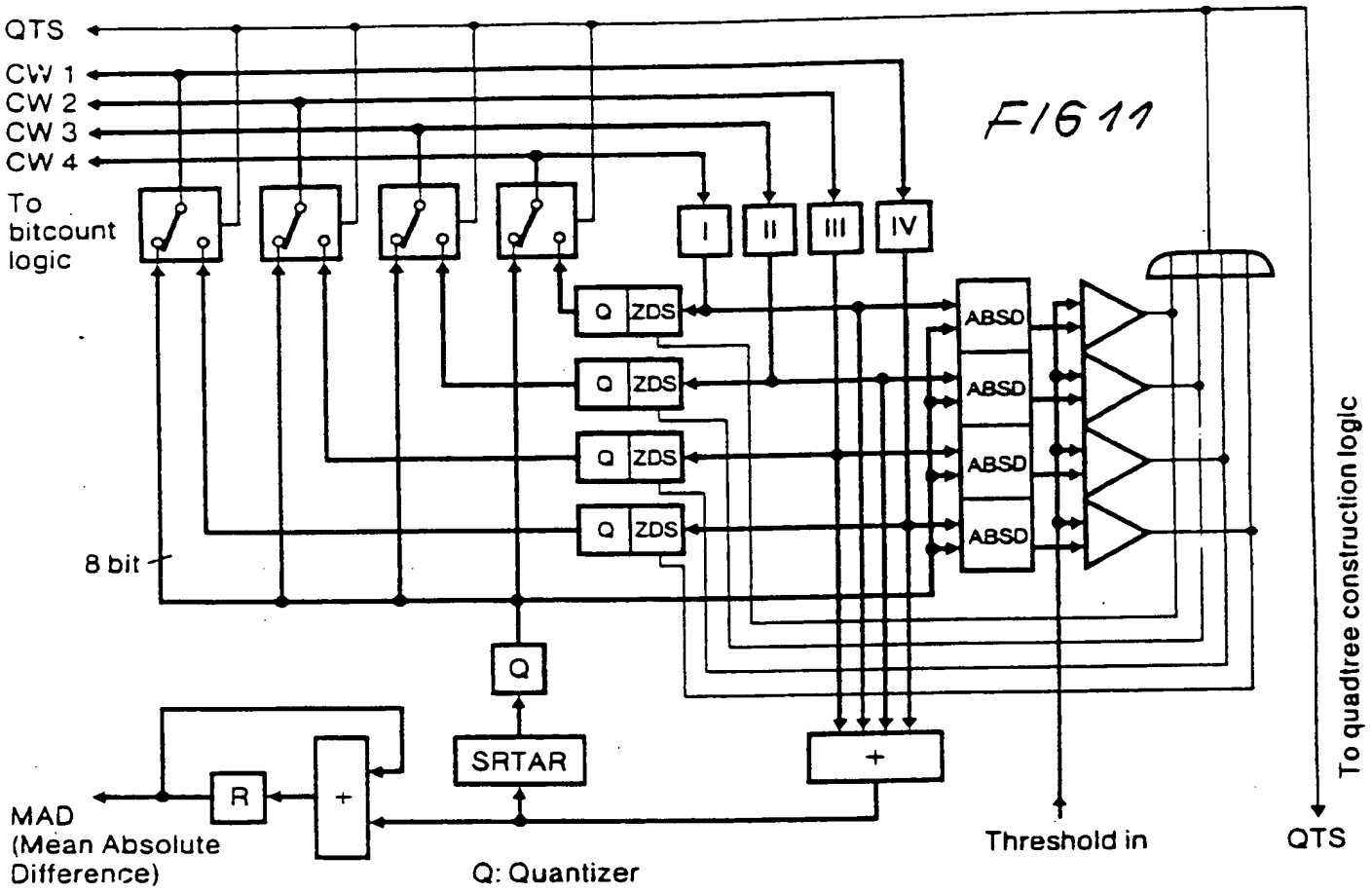
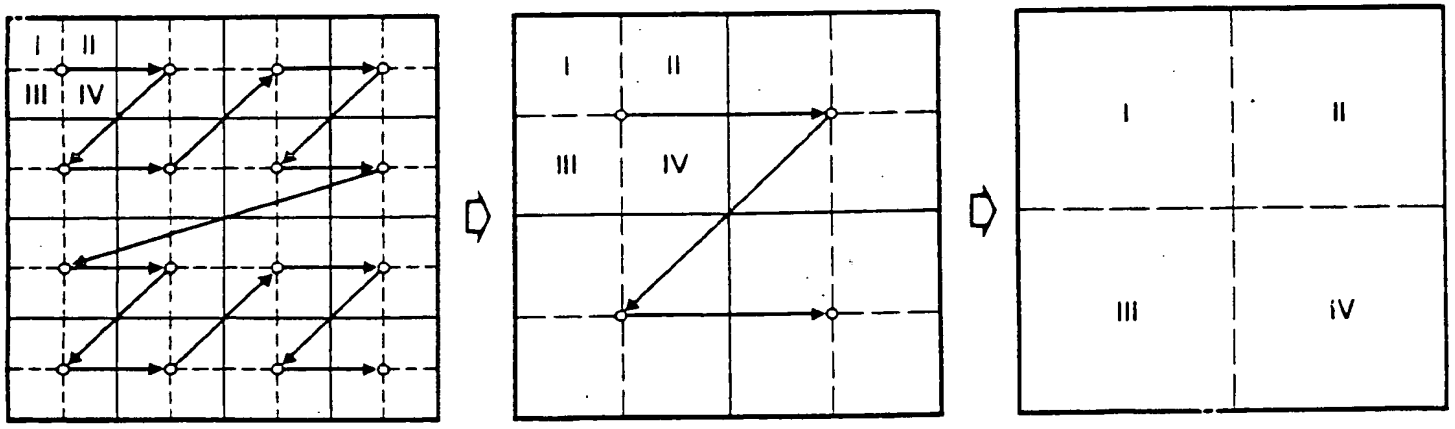


FIG 11



Schritt 1 (Level 1)

Entscheidung: 2*2 ⟷ 4*4

Schritt 2 (Level 2)

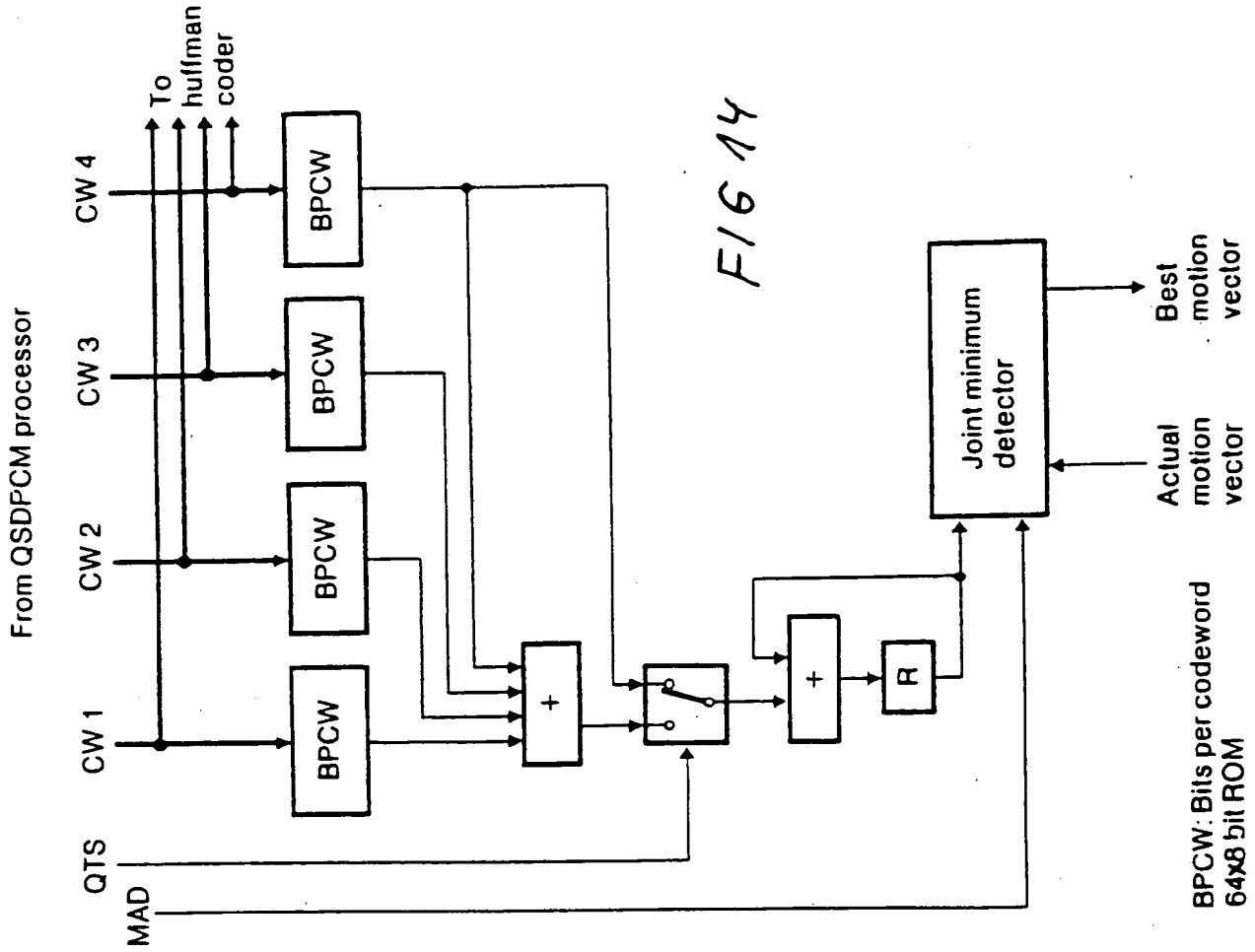
Entscheidung: 4*4 ⟷ 8*8

Schritt 3 (Level 3)

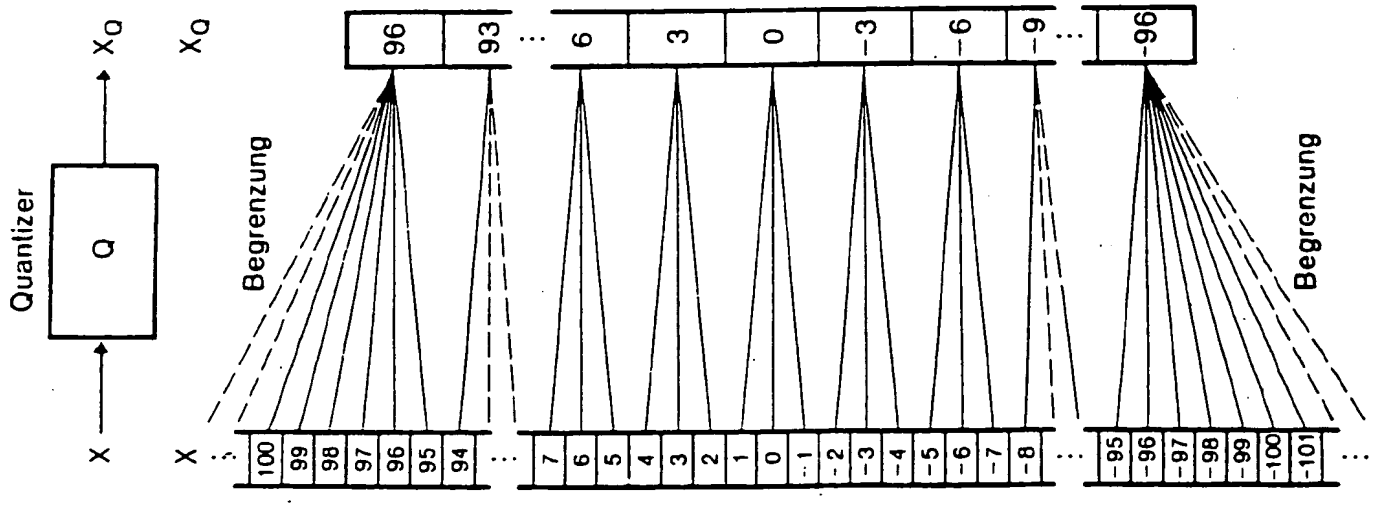
Entscheidung: 8*8 ⟷ 16*16

FIG 12

DE 37 33 038 C2
H 04 N 7/137
Veröffentlichungstag: 5. Januar 1994



F16 14



F16 13

Nummer:
Int. Cl.[5]:
Veröffentlichungstag:
DE 37 33 038 C2
H 04 N 7/137
5. Januar 1994

FIG 15

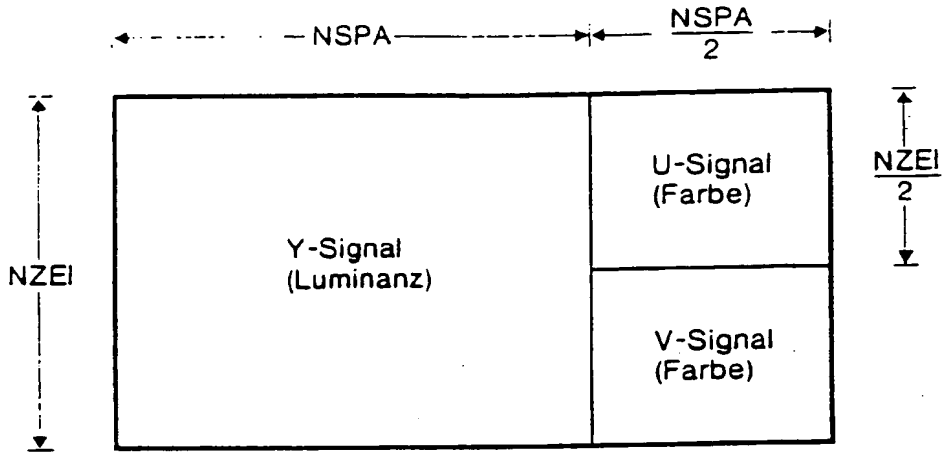
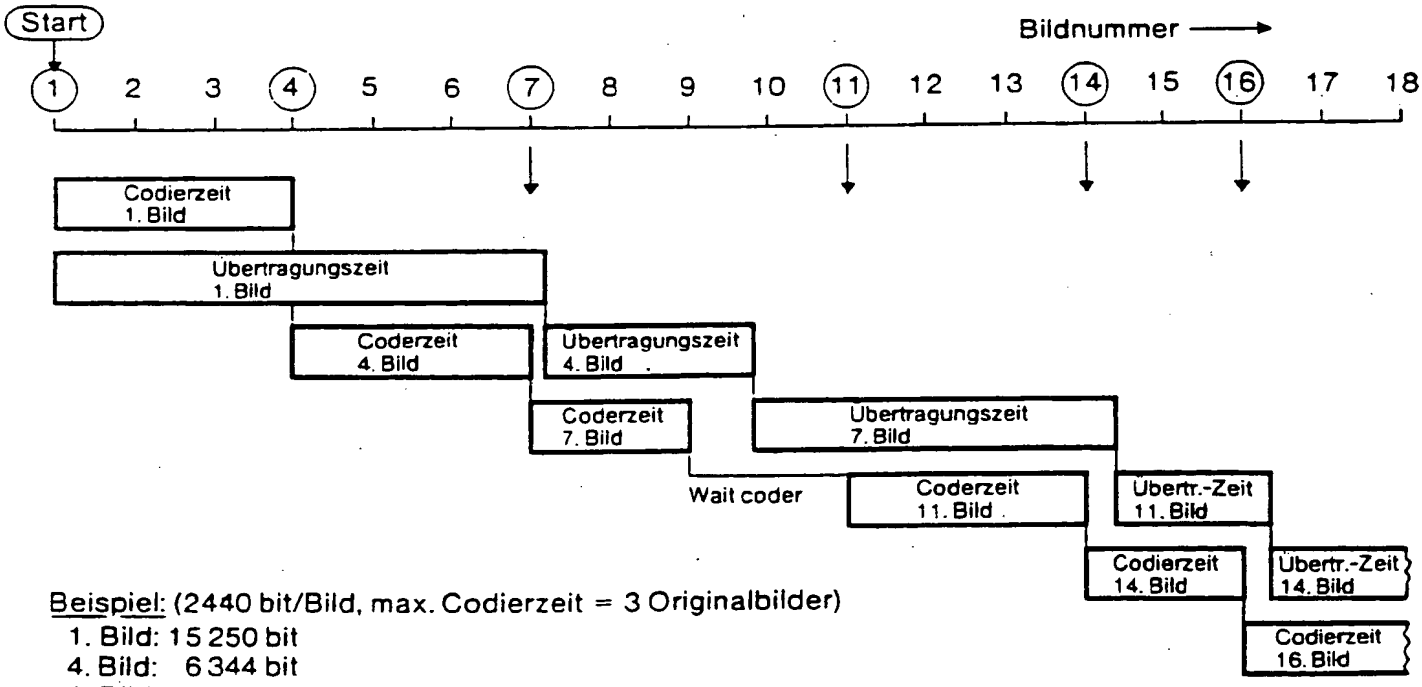


FIG 16 Zeitdiagramm variable frame rate



Beispiel: (2440 bit/Bild, max. Codierzeit = 3 Originalbilder)

1. Bild: 15 250 bit
4. Bild: 6 344 bit
7. Bild: 11 386 bit
11. Bild: 4 554 bit
14. Bild: ...

Nummer: DE 37 33 038 C2
Int. Cl.5: H 04 N 7/137
Veröffentlichungstag: 5. Januar 1994

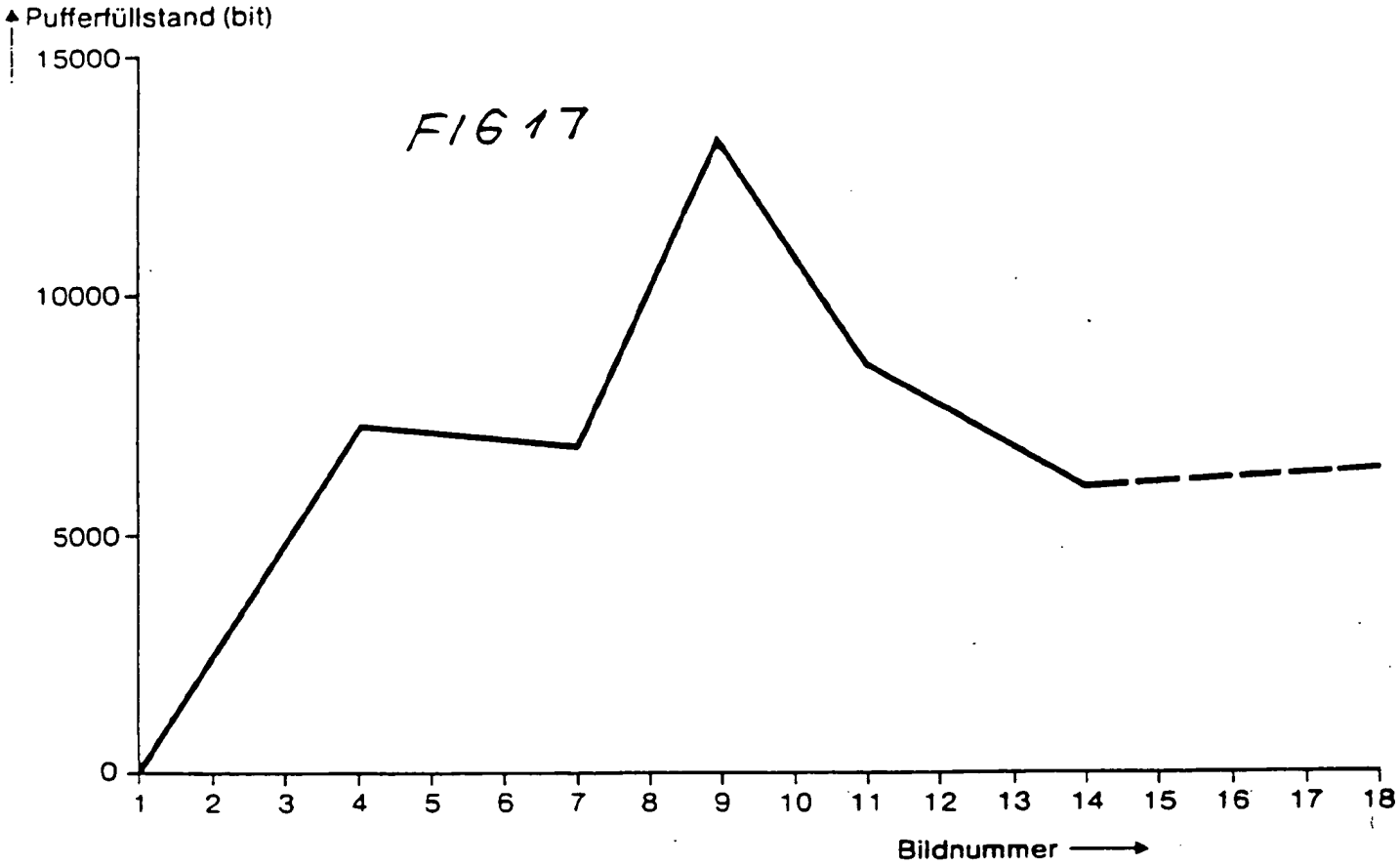


FIG 17

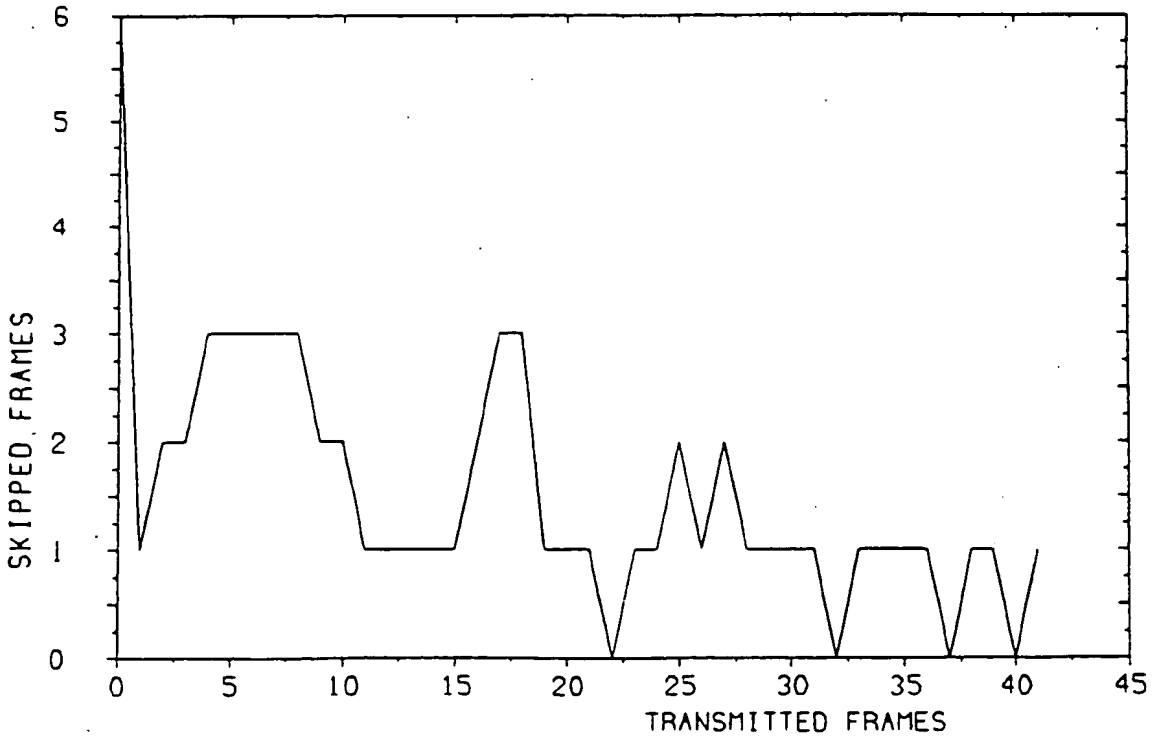


FIG 18

Nummer:
Int. Cl.[5]:
Veröffentlichungstag:
DE 37 33 038 C2
H 04 N 7/137
5. Januar 1994

FIG 19

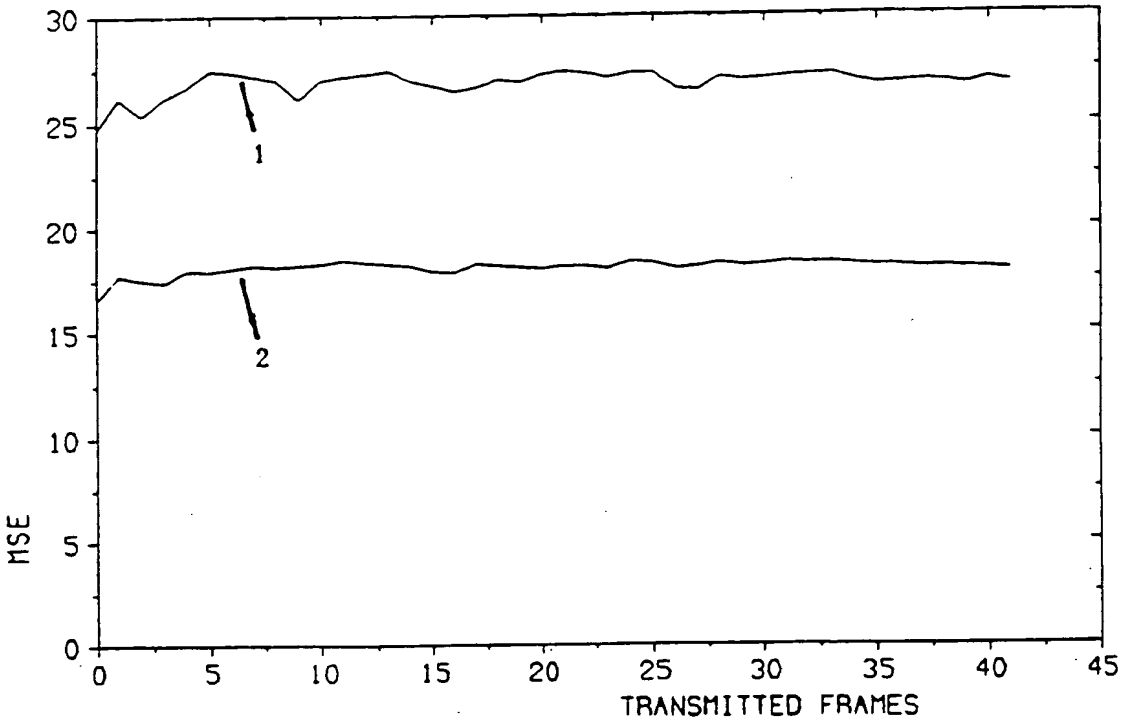
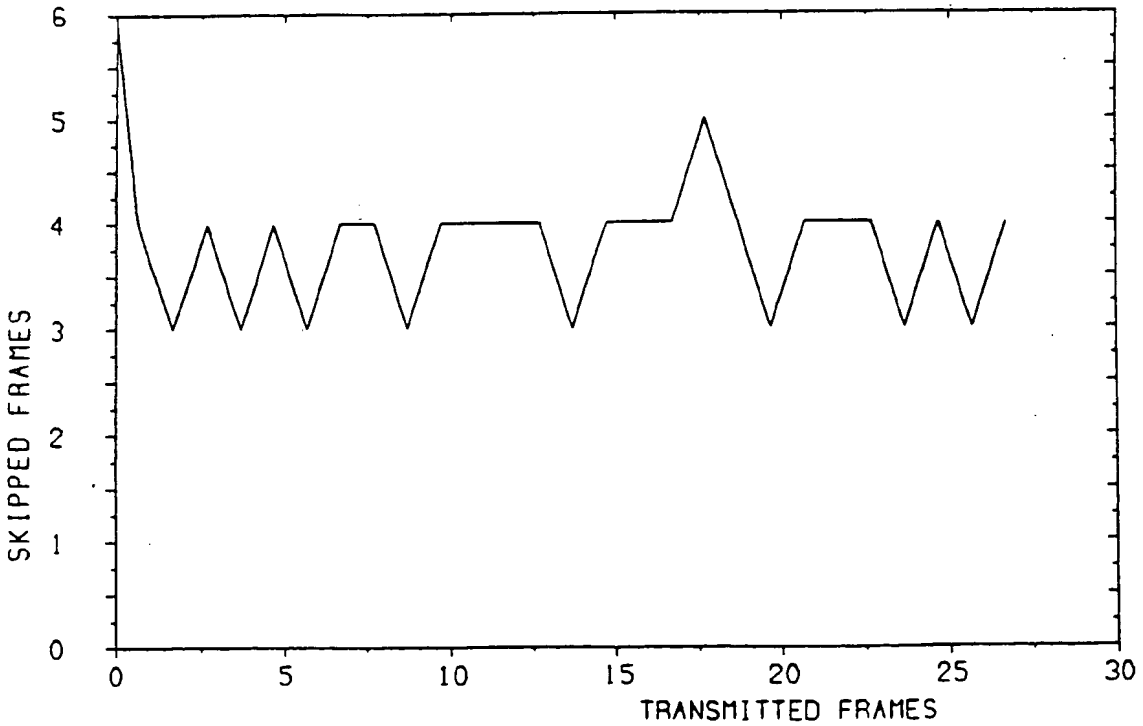


FIG 20



[redacted]mer: DE 37 33 038 C2
[redacted]Cl.[5]: H 04 N 7/137
Veröffentlichungstag: 5. Januar 1994

FIG 21

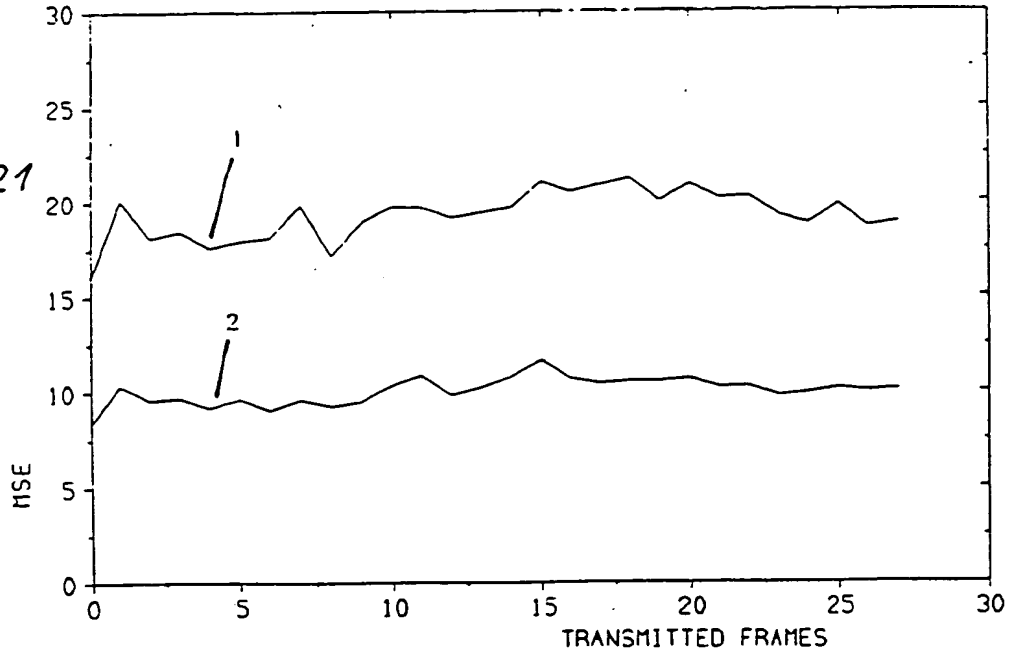


FIG 22

FIG 23

Nummer: DE 37 33 038 C2
Int. Cl.[5]: H 04 N 7/137
Veröffentlichungstag: 5. Januar 1994

FIG 24

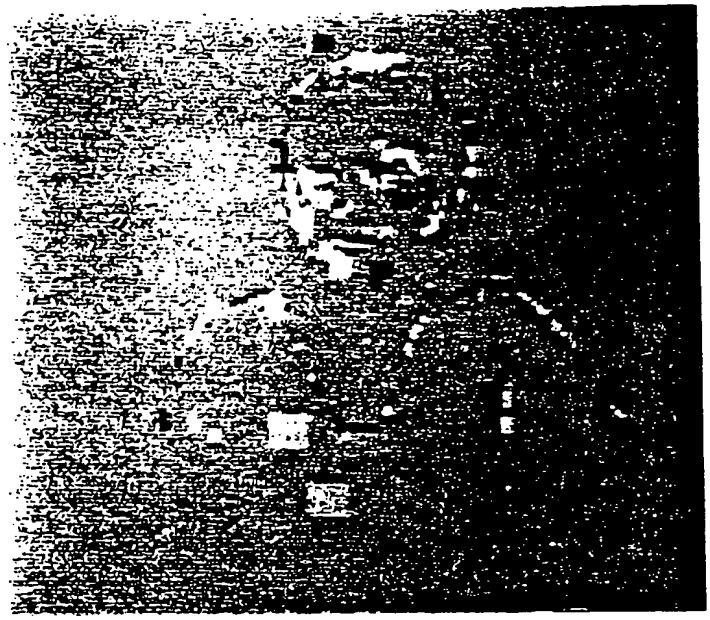
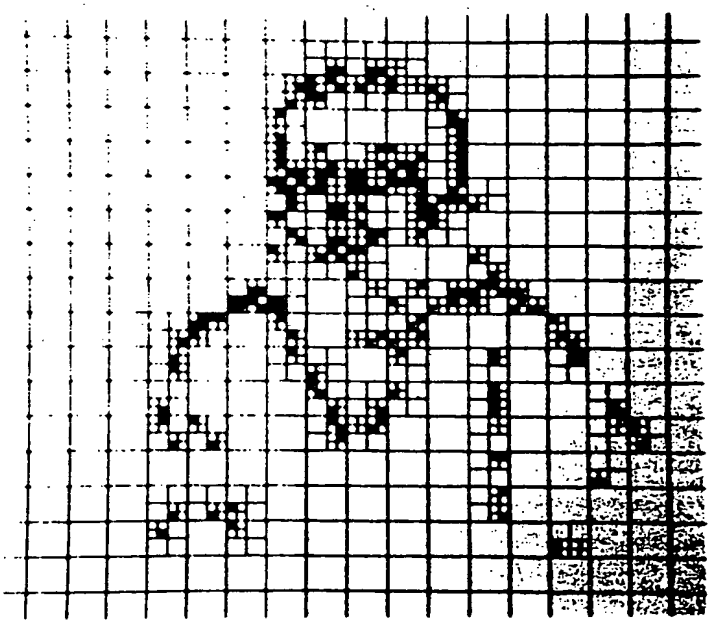


FIG 25



...mer: DE 37 33 038 C2
Int. Cl.[5]: H 04 N 7/137
Veröffentlichungstag: 5. Januar 1994

FIG 26

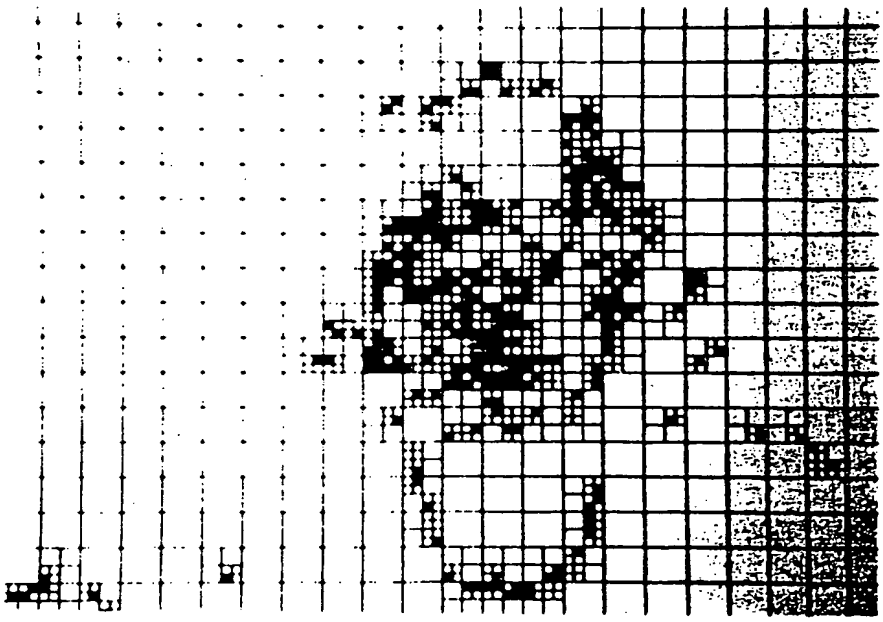
FIG 27

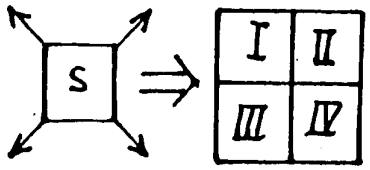
Nummer:
Int. Cl.[5]:
Veröffentlichungstag:

DE 37 33 038 C2
H 04 N 7/137
5. Januar 1994

FIG 28

FIG 29



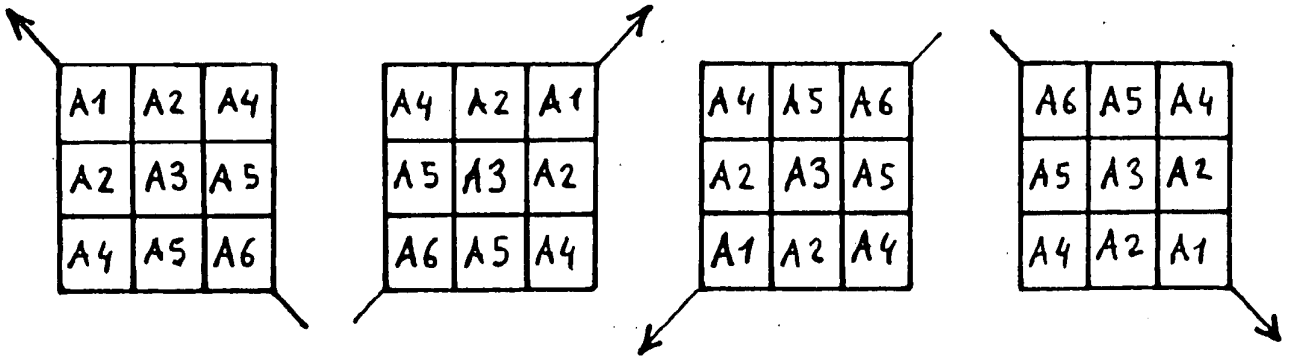


Berechnung der vier Einzelpixel I, II, III, IV durch Interpolation aus dem Pixel S.

FIG 30

FILTER ZUR REKONSTRUKTION VON:

PIXEL I PIXEL II PIXEL III PIXEL IV



KOEFFIZIENTENMASKE MIT KOEFFIZIENTEN A1, A2, A3, A4, A5, A6

DE 37 33 038 C2
H 04 N 7/137
Veröffentlichungstag: 5. Januar 1994

Numm
Int. Cl.
Veröffentlichungstag: 5. Januar 1994

DE 37 33 038 C2
H 04 N 7/137

FIG 31